VŠB – Technická Univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Testování Nativních XML databází pro velké kolekce XML dokumentů
Testing of Native XML Databases with Large Data Volumes

2012 Tomáš Bohuněk

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Bohuněk**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Testování Nativních XML databází pro velké kolekce XML dokumentů
Testing of Native XML Databases with Large Data Volumes

Zásady pro vypracování:

Hlavním úkolem nativních XML databází je uložit XML dokumenty a dotazovat. Náplní této práce bude testovaní XML databází pro velké kolekce XML dat.

Úkoly:
1. Nastudujte problematiku nativních XML databází.
2. Vyberte co největší počet XML databází.
3. Proveďte testy pro různé kolekce XML dat.
4. Porovnejte rychlost vykonávání dotazů.
5. Výsledky experimentů vyhodnoťte.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Chovanec**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta: Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 17.8. 2012 ……

Zadání:
Hlavním úkolem nativních XML databází je uložit XML dokumenty a dotazovat. Náplní této práce bude testovaní XML databázi pro velké kolekce XML dat.

Úkoly:
1. Nastudujte problematiku nativních XML databází.
2. Vyberte co největší počet XML databází.
3. Proveďte testy pro různé kolekce XML dat.
4. Porovnejte rychlost vykonávání dotazů.
5. Výsledky experimentů vyhodnoťte.

Poděkování: Rád bych tímto poděkoval vedoucímu práce Ing. Peteru Chovancovi, který tematickými komentáři přispěl k vypracování práce. Dále bych rád poděkoval společnosti Syncro Soft, která poskytnutím volné licence na svůj produkt oXygen XML Editor umožnila velmi rychlé zpracování XML dokumentů (generování, testovací dotazy apod.).

Abstrakt: Úkolem této práce je vybrání vhodných volně dostupných Nativních XML databází a provedení základních testů na těchto databázích. Výstupem této práce je doporučení konkrétní databáze pro konkrétní použití. Byly vybrány nejvyužívanější a nejdostupnější databáze s tím, že důraz byl kladen na dostupnost podpory při implementaci testů (internetová fóra) a na podporu dotazovacích jazyků.

Klíčová slova: XML dokument, XML databáze, kolekce, round-tripping, aktualizace, dotaz, XQuery, XUpdate

Abstract: In the beggining of this bachelor's work we had to choose Open-Source Native XML databases for base testing on these databases. Result of this work could be a recommending some Native XML database. The most used and most available were eXist, BaseX, Sedna and MonetDB. The biggest importance for choosing were maintenance for implementing the tests (web) and supporting query languages.

Keywords: XML document, XML database, round-tripping, collection, update, query, XQuery, XUpdate

Obsah

1. Úvod

Formát XML se od doby, kdy byl konsorciem W3C[1] vyvinut a standardizován, stal jedním z nejpoužívanějších formátů pro výměnu dat mezi aplikacemi a pro publikování dokumentů, u kterých popisuje strukturu z hlediska obsahu jednotlivých částí a nezabývá se vzhledem. XML formát vznikl z důvodů potřeby existence jednoduchého otevřeného formátu, který není úzce spjat s konkrétní platformou[2] či proprietární technologií. XML formát pro ukládání dat v souborech používají v praxi systémy jednoduchými webovými stránkami počínaje a složitými aplikačními a databázovými servery konče.
Cílem této bakalářské práce je otestovat možnosti použití nativních XML databází pro ukládání a nakládání s velkým objemem dat. Jedná se o zavedení nového ne příliš používaného (např. ve srovnání s relačními[3] databázemi) způsobu ukládání dat.
V druhé části se zaměřím na představení XML databází jako takových a jejich možností. V třetí části představím 4 vybrané databáze, které byly použity pro jednotlivé testování. Ve čtvrté kapitole se zaměřím na popis operací, které byly nad databázemi spouštěny. V páté části představím, jaké vlastnosti jsem u jednotlivých databází měřil a jakým způsobem došlo k nakládání s výsledky měření. Pro samotné testování byla použita vlastní aplikace vytvořená v programovacím jazyce Java.
V poslední kapitole zhodnotím výsledky testů a nastíním možnosti použití jednotlivých databází v praxi.

[1] World Wide Web Consorcium – Mezinárodní společenství pro standardizaci World Wide Web
[2] Platforma – operační systém, prostředí vyvíjené aplikace
[3] Relační databáze – databáze založená na relačním modelu

2. XML databáze

Tato kapitola se zaměří na představení XML databází [1]. Stručně si popíšeme oba dva druhy XML dokumentu a také si rozdělíme XML databáze na rovněž dva druhy. V třetí části této kapitoly si stručně představíme několik dostupných Nativních XML databází [2] a popíšeme si způsob práce s kolekcemi potažmo dokumenty. Všeobecně lze říci, že XML soubory lze ukládat přímo v souborovém systému ovšem bez jakýchkoliv optimalizací, na které jsme zvyklí například u relačních databází, ve formě indexace[4], více uživatelského přístupu[5], transakčních manažerů[6], bezpečnosti a podobně. Logicky zde vzniká potřeba použití nějakého databázového systému, který tyto služby dokáže nabídnout.

2.1 XML dokument

Jedná se textový soubor, který ovšem musí splňovat náležitosti, co se struktury týče. Data v tomto souboru jsou navíc popsána pomocí definovaného systému značek[7].

2.1.1. Dokumentově orientovaný XML dokument

Tento dokument [3] můžeme popsat tak, že takový dokument má nepravidelnou strukturu, hodně smíšeného obsahu. Takovýto dokument je obvykle vytvářen ručně uživatelem. Významnou část dokumentu tvoří tzv. elementy se smíšeným obsahem (kombinace textu a vnořených elementů). V takovém dokumentu bývají velmi často ukládány knihy, emaily, brožury, reklamy, články v blozích apod. Stručně tedy lze říci, že tyto dokumenty jsou určené pro čtení a zpracování lidmi.

Příkladem takového typu dokumentu může být například podobná HTML stránka:

[4] Indexace – metoda optimalizace dat
[5] Více uživatelský přístup – v jeden okamžik s daty pracuje více uživatelů
[6] Transakční manažer – systém řízení transakcí
[7] značka (tag) – název elementu

```
<html>
<body> Včera večer jsem navštívil malé kino na rohu
Sokolovské v Praze, které se jmenovalo Atlas. Ve foyer
kina se nachází kavárnička s vinotékou a velmi příjemnou
obsluhou. Shlédnutí kutlovního snímku <b>Řeč krále</b> v
příjemné atmosféře ve mne zanechalo neuvěřitelný kulturní
zážitek, který bych rád touto cestou doporučil.
       </body>
</html>
```

2.1.2. Datově orientovaný XML dokument

Zpravidla se jedná o dokument [3], který je naopak určen zejména pro strojové zpracování, neboť musí mít pevně definovanou pravidelnou strukturu. Tyto soubory jsou přednostně používány pro jakýkoliv transport mezi aplikacemi. Dokumenty mají nulový nebo velmi malý počet elementů se smíšeným obsahem. Tyto dokumenty jsou obecně používány pro přenos faktur, vědeckých dat, jízdních řádů, telefonních seznamů, katalogů, zprávy webových služeb[9], osobní záznamy, objednávky a jiné.

Příklad konfiguračního souboru web.xml (aplikační server[8] JBoss[4]):

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
    <context-param>
        <param-name>org.richfaces.SKIN</param-name>
        <param-value>jboss-console</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-
class>javax.faces.webapp.FacesServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.seam</url-pattern>
    </servlet-mapping>
</web-app>
```

8 Aplikační server – vrstva mezi operačním systémem a aplikací
9 webová služba – umožňuje komunikaci dvou aplikací v sítí (přijímání/odesílání zpráv)

2.2. Rozdělení XML databází

V úvodní části dojde k vysvětlení samotného termínu „XML databáze“. Celou problematikou se zabývá iniciativa kolem XML:DB [5]. Tato iniciativa si vytýčila řešit problém ohledně formulace a specifikace způsoby nakládání s daty v XML databázích. Dle iniciativy XML:DB existují tři různé druhy XML databází. První jsou XML Enabled Databáze (XEDB) [6], druhou skupinou jsou Nativní XML databáze (NXD) [2]. Třetí typ se nazývá Hybrid XML Database (HXD) [3].

2.2.1. XML Enabled Database

V tomto případě jde o databáze, které jsou rozšířeny o XML mapovací vrstvu, která je poskytována přímo prodejcem či třetí stranou. Tato mapovací vrstva se stará o ukládání a načítání XML dat. Data, která jsou mapována do databáze, jsou namapována na specifický formát a původní XML meta-data[10] a struktury mohou být ztraceny.
V databázích tohoto typu není zaručeno, že získaná data z databáze budou odpovídat původní formě. Může k tomu dojít manipulací s daty prostřednictvím různých XML technologií jako jsou např. XPath [7], XSLT [7], DOM[11] nebo SAX[12], ale i SQL[13]. Tudíž je základní jednotka uložení v XEDB závislá na zvolené implementaci.

Do této kategorie XML řešení spadá řešení od firem jako jsou Oracle, Microsoft a mnoho dalších.

2.2.2. Nativní XML databáze

- Definuje (logický) model pro XML dokumenty, podle kterého je ukládá a načítá. Je nutností, aby model obsahoval zejména elementy, atributy, PCDATA[14] a informace o pořadí. Příkladem takového modelu je XPath datový model, XML Infoset[15] a nebo modely vycházející z DOM a událostí v SAX 1.0.
- XML dokument je použit jako základní (logická) jednotka úložiště, stejně tak jako je použit řádek jako základní jednotka úložiště v tabulce relační databázi.

10 Meta-data – strukturovaná data o datech
11 Document Object Model – objektově orientovaná reprezentace XML nebo HTML dokumentu
12 Simple API for XML – podpora sériového přístupu ke XML datům
13 Structured Query Language – standardizovaný dotazovací jazyk
14 Parsed Character Data – definice struktury dat
15 XML Information Set – W3C specifikace popisující model dat

- Odpadá nutnost mít zvláštní fyzické úložiště základního modelu. Databáze může tedy být postavena nad relační, hierarchickou nebo objektově orientovanou databází, nebo může být použit proprietární formát úložiště jako jsou například indexované nebo komprimované soubory.

2.2.3. Hybridní XML databáze

Do této databáze lze zahrnout databáze, které mohou být považovány jak za nativní, tak za databáze XML Enabled. Zařazení záleží na požadavcích aplikace, která databázi využívá. Mezi tyto databáze bývá zařazována databáze Ozone [8], jejíž vývoj a podpora byla ukončena v roce 2009.

2.3. Nativní XML databáze

XML databáze [2] jsou primárně založeny na datovém formátu XML a k němu přidružených standardů.
Tyto databáze pro svoji činnost využívají tzv. **Kolekcí**. Kolekcí je v tomto směru myšlen soubor určitým způsobem související s XML dokumenty. Kolekce plní úlohu tabulky v relační databázi. Ovšem podpora v relačních databázích je závislá na jejích výrobcích (vývojářích). Kolekce mohou být závislé na XML schématu (Schema dependent).
Standardem pro dotazování v NXD se stal **jazyk** XQuery [9]. Na rozdíl od dotazovacího jazyka SQL se ovšem zaměřuje na samotné dotazování výběru dat z databáze. Nenabízí tudíž dotazy typu INSERT, UPDATE, DELETE. Pro tyto účely je nutné zvolit nějaký aktualizační jazyk. Nejčastěji používaným se stal jazyk XUpdate [10], který vyvinula iniciativa XML:DB. Dalším zástupcem aktualizačních jazyků je např. XQuery Update [11], který vyvíjí a podporuje konsorcium W3C. Aktualizační jazyk nabízí operace přidání, aktualizace a mazání libovolných částí XML stromu. V případě XUpdate se jedná o samostatný jazyk, XQuery Update je rozšíření jazyku XQuery.
NXD v neposlední řadě nabízí i možnost **indexace** XML dat. Mezi základní způsoby indexace dat patří:

- Strukturální indexy – takové indexy jsou pro použití vyhledávání podle jména a pozice elementu nebo atributu
- Hodnotové indexy – tyto indexy je lépe použít v případě, že je očekáváno vyhledávání podle hodnoty elementu

- Full-textové indexy – jak již název napovídá, jedná se o indexy vhodné pro použití hledání v obsahu. Příkladem dotazu může být – vyhledej všechny elementy obsahující určité slovo.

V způsobech indexování jsou mezi databázemi rozdíly. Rozdíly jsou zejména mezi druhy podporovaných indexovacích metod a také v jejich konfiguraci. Většina takových databází podporuje primárně strukturální a hodnotový index. Jiné databáze ovšem poskytují i jiné možnosti indexování. Proto je důležité se před použitím seznámit s danou databází a jejími možnostmi.
Co se **aplikačního rozhraní** pro práci s NXD týče, nabízí ve většině případů každá databáze vlastní rozhraní. Existují ovšem i obecná rozhraní (podobná jako ODBC[16] pro relační databáze), z kterých je možno vycházet. Popisem těchto rozhraní se ovšem v této práci zabývat nebudeme, protože téma samo o sobě je poměrně rozsáhlé.
V další části se bude zabývat architekturou NXD. Architekturou NXD je myšlen přístup, jakým databáze ukládá data. Základní rozdělení je na textovou architekturu a na architekturu založenou na modelu dat.

2.3.1. Textová architektura

Pokud NXD ukládají XML dokumenty přímo v textové formě, je taková databáze označována jako textově založená (text-based) NXD. Tímto způsobem ukládání dat je zaručena vysoká úroveň round-trippingu. Samotná realizace uložení může být různá: uložení XML souboru v souborovém systému nebo CLOB[17] (BLOB[18]) poli relační databáze, nebo vlastní textový formát dané databáze.
Základní vlastností této architektury je možnost použití indexování, které při dotazování do databáze a získávání celých dokumentů nebo jejich částí značně urychluje vykonání dotazu. V tomto případě stačí pomocí indexu získat pozici dat v dokumentu a následně jej přečíst. Odpadá tedy nutnost spojení menších jednotek dat, jako je tomu například v případě relačních databází. Taková architektura vybízí k využití v aplikacích, u kterých převládá jeden pohled na data.

16 Open Database Connectivity – standartizovaný systém pro přístup k databázovým systémům
17 Character Large Object – datový typ pro ukládání velkého množství dat
18 Binary Large Object – datový typ určený pro ukládání velkého množství binárních dat

2.3.2. Architektura založená na modelu dat

Databáze založené na datovém modelu (model-based) pracují tak, že ukládají XML data dle vlastního datového modelu, který se byl uložen pomocí jiné databáze nebo jiného vlastního datového formátu. Tyto databáze (založené na jiném databázovém systému) bývají z pohledu výkonnostního na úrovni použitého systému. Taková NXD oproti tomu nabízí velkou volnost ve způsobu, jakým lze XML databázi na jiné platformě postavit. Tento způsob je ovlivněn hlavně výběrem datového modelu a způsobu mapování použité databáze. V případě použití vlastního formátu uložení dat, jsou často mnohem rychleji vyřizovány dotazy se strukturou ekvivalentní s uloženým dokumentem jako v případě textově orientovaných databází.

2.3.3. Dostupné Nativní XML databáze

Tato část obsahuje výběr světově nejpoužívanějších NXD databází [2]. Komerční databází je myšleno databáze, která je použita pro účely finančního zisku (ve firmách). Open-Source je na druhou stranu databáze určená pro studium, zkoumání a testování chování těchto databází.

2.3.3.1. Volně dostupné databáze

produkt	vývojář	typ db
BaseX	Univerzita Konstanz	proprietární
Berkeley DB XML	Oracle	klíč-hodnota
eXist	Wolfgang Meier	proprietární
MonetDB/XQuery	CWI Database Group	proprietární
Xindice	Apache Software Foundation	proprietární
Sedna XML DBMS	ISP RAS MODIS	proprietární
dbXML	dbXML Group	proprietární

BaseX– tato databáze [12] je postavena na architektuře klient-server a nabízí tak vysoký výkon. Databáze nabízí podporu víceuživatelského přístupu, tudíž i podporu souběžnosti v databázi. Dle internetových zdrojů je databáze vhodná pro ukládání velmi velkých XML dokumentů. Díky těmto aspektům se databáze jeví jako vhodný kandidát pro testování, které je tématem této práce.
Berkeley DB XML – za vývojem a podporou této databáze stojí známý databázový gigant - společnost Oracle. Databáze [13] nabízí podobné funkce jako výše jmenované databáze. Drobnosti při nasazení databáze na testovacím stroji byly příčinou vyřazení této databáze ze samotného testování.

eXist – databáze [14] nabízí veškerou funkčnost, které je třeba pro provedení našeho testování. Databáze nabízí možnost víceuživatelského přístupu, správu uživatelů, souběžný systém dotazování apod. Přednost této databáze je z mého pohledu spatřována v dostupnosti kvalitního klienta, který je schopný obsloužit většinu uživatelských požadavků (přidání nové kolekce, smazání kolekce, přidání/mazání dokumentu z kolekce, dotazování apod.). Díky tomu byla tato databáze vybrána pro otestování.
MonetDB / XQuery – poskytuje [15] většinu funkcí, na které jsme zvyklí z běžných databází. Je zajištěna podpora jazyka XQuery, včetně modulů, transakce pomocí XQuery Update Facility, uživatelem definovaných funkcí apod. Dle internetových zdrojů je systém vhodný pro použití zpracování velkých XML kolekcí. Proto byl tento systém vybrán k otestování.
Xindice – tato databáze [16] je vhodná pro ukládání velmi složitých struktur XML, které by bylo těžké mapovat do strukturovaných databází. Vývoj této databáze je ovšem prozatím ukončen. Databázi je také nutno provozovat jako aplikaci webové serveru. Tyto dva aspekty byly odrazující pro další testování.
Sedna – jedná se databázi [17], která opět nabízí vysokou dostupnost potřebných funkcí. Správou uživatelů, kteří s databází chtějí pracovat, počínaje, možností indexování a podporou transakcí a dotazovacích jazyků konče.
dbXML – tato databáze [18] je charakterizována schopností uchovávání a indexování kolekcí XML dokumentů a vysoce efektivním dotazováním, schopností zpracovávat transakce a vyhledávání apod. Databáze také nabízí rozšíření o obchodní logiku ve formě skriptů, tříd a triggerů. Databáze ovšem v době testování nenabízela rozumné rozhraní pro použití v programovacím jazyce Java. A proto bylo upuštěno od provádění testů této databáze.

2.3.3.2. Komerční databáze

Po negativních reakcích oslovených vývojových týmů a týmů podpory dvou náhodně vybraných komerčních databází nebylo možné tyto databáze otestovat.

produkt	vývojář	typ db
MarkLogic Server	Mark Logic Corp.	proprietární
Qizx	XMLMind	proprietární
Tamino	Software AG	proprietární
Sonic XML Server	Sonic Software	objektově-orientovaná

2.3.3.3. Nativní XML databáze versus relační databáze

Většina relačních databází(Oracle Database, Microsoft SQL Server, IBM DB2 a jiné) v současnosti nabízí možnost využití datových formátů založených na XML. Porovnání relačních databází a nativních XML databází bohužel není příliš jednoznačné. Velmi záleží na struktuře dat, na aplikační logice vyvíjené aplikace, na platformě, na které data budou spravovány apod.
Klíčovou otázkou tedy zůstává, jaký formát dat bude používat vyvíjená aplikace.
Otázkou porovnání Relačních databází oproti Nativním XML databázím se zabývá ve svém „Research reportu“ Anne Williams [19]. Ve své práci využívá relační databáze PostgreSQL a nativní XML databáze eXist. Jako zdrojová data použil 6 432 emailů, které uložil do databází a prováděl nad nimi dotazy, které následně porovnával. Ani zde není jednoznačně určeno, která databáze je vhodná použít. Například pro dotaz, kdy je prováděna selekce z databáze pomocí unikátního klíče musí relační databáze projít tři tabulky pomocí tzv. „full scan“[19] tabulek. To je pro databázi obecně časově náročná operace a v tomto případě dosahuje lepších výsledků databáze eXist. Selekce z databáze PostgreSQL trvala 5 028 milisekund. Databáze eXist provedla stejný dotaz za 3 621 milisekund.

Příklad dotaz 1

```
XQuery:
for $x in collection("spam")/message where $x/@id = 10000
return $x

SQL:
select * from tbl_message LEFT OUTER JOIN tbl_header on
(tbl_message.message_id = tbl_header.message_id) LEFT
OUTER JOIN
tbl_recipient on(tbl_message.message_id =
tbl_recipient.message_id)
where tbl_message.message_id = 10000;
```

19 full table scan – způsob průchodu celou tabulkou za účelem hledání dat

Pro jeden z nejpoužívanějších dotazů používaných v relačních databázích – selekce dat z jedné tabulky – hrají výsledky ve prospěch databáze PostgreSQL. Dotaz, který provede selekci dat z jedné tabulky byl proveden v databázi PostgreSQL za 529 milisekund a v databázi eXist za 25 432 milisekund.

Příklad dotaz 2

```
XQuery:
for $x in //recipient-list return $x//recipient-address

SQL: select * from tbl_recipient;
```

Nelze tedy jednoznačně říci, kterou databázi je vhodné použít. Vždy bude záležet na konkrétním případu použití.
Tomuto tématu se také věnuje jeden z hlavních vývojářů pracující ve společnosti IBM Matthias Nicola [20].

3. Vybrané nativní XML databáze

Byly vybrány 4 Nativní XML databáze eXist, BaseX, Sedna, MonetDB a to s ohledem zejména na možnost použití obecného aplikačního rozhraní ve vlastní testovací aplikaci. Dále byl brán ohled na intenzitu s jakou jsou dané databáze používány v praxi. V neposlední řadě bylo přihlédnuto na kvalitu dostupné dokumentace. Poslední kritérium podle kterého došlo k výběru těchto databází byla dostupná podpora ze strany dodavatele (vývojářská skupina). Byly vybrány Open-Source licencované databáze, neboť skupiny starající se o rozvoj a podporu dvou vybraných komerčních databází se k poptávce po uvolnění licence za účelem testování pro bakalářskou práci nevyjádřily. Tudíž nemohlo dojít k jejich otestování.

Provoz databází

databáze	verze	operační systém	implementace
eXist	1.4.0	Linux, Windows	server, servlet, embeded
BaseX	6.0	Linux, Windows	server
Sedna	3.5	Linux, Windows	server
MonetDB/XQuery	4.0	Linux, Windows	server

3.1. Databáze eXist

Databáze byla založena v roce 2000. S jejím vývojem začal vývojář Wolfgang Meier poté, co během letní dovolené přečetl nespočet odborných článků o účinných metod pro indexování XML. Projekt stále prochází dalším vývojem. Vedoucím projektu a hlavní vývojářem je stále Wolfgang Meier. Databáze byla původně zaměřena na dokumentově orientované XML soubory. Později byla doplněna o funkcionalitu nabízejí i snazší práci s datově orientovanými XML soubory.

Uložení dat: databázový server eXist pracuje s jednou databází, v které se může vyskytovat několik kolekcí. Databázový model s vysokou úrovní pro round-tripping nabízí vracení dat v podobě DOM stromu. Dokumenty jsou tedy uloženy v hierarchické struktuře kolekcí. Tato struktura je podobná struktuře adresářů v libovolném souborovém systému. Do kolekcí je možné ukládat libovolné dokumenty bez ohledu na schéma. Dané kolekci je ale možné schéma přiřadit. Podporovaná schémata jsou DTD, XML Schema, RelaxNG, Schematron, Namespace-based Validation Dispatching Language. Nastavení jako je

např. indexování apod. je vázané vždy na kolekci dokumentů. Počet kolekcí a velikost ukládaného XML dokumentu není omezena.
Práce s daty: dotazy na výběr dat z databáze je možno provádět pomocí dotazovacího jazyka XQuery. Dotazovat lze jeden dokument nebo celou kolekci. Dotazování je možno provádět rekurzivně.
Indexování: celé indexování je v této databázi řešeno modulárně – tzn. Při čtení vstupního XML dokumentu nebo při další manipulaci s ním jsou generovány události zasílané každému indexovacímu modulu. Tento modul na základě těchto událostí udržuje index. K indexování dochází v rámci jedné kolekce. Není tedy možné nastavit indexování pro každý XML dokument zvlášť. V dotazech jsou automaticky zohledňovány. Je ovšem třeba pokládat dotazy, které daný index opravdu využívají. Základní moduly jsou obsaženy v dodávce databáze. Je ovšem možné i vytvářet vlastní .
Bezpečnost: v databázi je možné vytvářet jednotlivé uživatele, které mohou mít různou úroveň oprávnění. Uživatelé v databázi jsou rozděleni do skupin, mohou mít domovskou kolekci a přístupová práva jsou přiřazena vlastníkovi, skupině.
Další možnosti: transakční systém zatím není možno z pozice uživatele databáze ovládat. Transakční systém je určen pouze pro obnovu databáze po kolizi. Nutno ještě zmínit, že databáze podporuje tzv. triggery (např. když se v dokumentu něco změní na jednou místě, automaticky to databáze může změnit na druhém místě, bez zásahu uživatele – podobně jako u triggerů z relačních databází.).
Nasazení: databázi eXist je možno provozovat jako samostatný server, část nějakého Java Enterprise Edition kontejneru[20] (Apache Tomcat, JBoss, Oracle Weblogic a jiné) a nebo jako součást vlastní aplikace (embeded). Databáze podporuje tato aplikační rozhraní: REST/HTPP, XML-RPC, SOAP. Za hlavní aplikační rozhraní je označeno rozhraní dodávané XML:DB.

3.2. Databáze BaseX

Tato databáze byla poprvé představena roku 2007. S jejím vývojem začal Christian Grün na Univerzity of Konstanz v roce 2005. Tato databáze je poskytována jako Open-Source. Systém je nezávislý na platformě. Celá databáze je napsána v jazyce Java a pro ukládání XML dat používá vlastní formát.

20 kontejner – část serveru starající se o úplné aplikační prostředí

Uložení dat: soubory tabulek obsahují informace o uzlech XML dokumentu. XML dokumenty jsou ukládány do kolekcí (v případě BaseX označované za databáze). Do kolekcí je možno ukládat libovolné množství XML dokumentů. Podporované datové formáty pro ukládání jsou: XML, HTML, CSV, Text a binární data.
Práce s daty: pro dotazování nad daty je standardně podporován jazyk XPath a XQuery. Pro aktualizace je podporován jazyk XQuery Update. Pro full-text dotazování je zavedena podpora jazyka XQuery Full Text. Dotazovat data v kolekcích je možné pomocí XQuery funkcí `fn:doc()` a `fn:collection()`.
Indexování: standardně jsou podporovány standardní textové a atributové indexy. Je zajištěna podpora indexů využívajících se pro full-textové vyhledávání. Dále je podporován index cest, který slouží k urychlení nalezení umístění v uzlu ve stromové struktuře.
Bezpečnost: oblast bezpečnosti je zajištěna systémem uživatelských práv. Na vrcholu v hierarchii uživatelů je uživatel `admin`, který má práva CREATE, WRITE, READ, NONE a další uživatelé v hierarchii mají práva WRITE, READ, NONE.
Další možnosti: Databáze nabízí vypracované GUI (Graphical User Interface), který umožnuje uživatelům interaktivní přístup ke kolekcím a dokumentům. Podporovaná aplikační rozhraní jsou XQJ, XML:DB a pro rozhraní jazyka Java má BaseX vlastní rozhraní.
Nasazení: BaseX je možno provozovat jako samostatný server nebo jako aplikaci typu klient-server.

3.3. Databáze Sedna

První verze databáze Sedna se objevila roku 2003. Jejím vývojem se zabývá oddělení MODIS Ústavu pro systémové programování Ruské akademie věd. Hlavním úkolem tohoto oddělení bylo vytvoření nativní XML databáze, která bude nabízet většinu běžných databázových vlastností jako je transakční manažer, aktualizace dat, možnosti v oblasti bezpečnosti a jiné.
Uložení dat: databáze nabízí přístup k několika databázím současně. Dokumenty jsou ukládány samostatně nebo jsou součástí kolekce. Pokud je kolekci přiřazeno XML schéma, je před vkládáním a aktualizací provedena validace XML dokumentu. Pokud je přiřazeno více schémat, je zároveň požadována úspěšná validace alespoň jednoho ze schémat.

Práce s daty: standardně je podporován dotazovací jazyk XQuery. Pro aktualizaci dat jazyk XQuery Update. Data lze vybírat jak z jednoho dokumentu, tak z celé kolekce dat, nebo z určitého dokumentu dané kolekce.
Indexování: indexovaní hodnot elementů a atributů je zajištěno dle datových typů převzatých z XML Schema XML dokumentů nebo kolekcí. Při klasickém dotazování nejsou indexy použity. Pro jejich začlenění je nutno použít speciální funkce `index-scan()`, případně `index-scan-between()` pro rozsahový dotaz. Tyto funkce se v XQuery dotazech chovají podobně jako funkce `doc()` – vrací uspořádanou množinu uzlů podle dotazovacích kritérií. Příprava indexů pro libovolné dotazy není možná, protože struktura pokládaných dotazů je určující pro návrh indexů. Databáze Sedna nabízí i podporu indexů pro full-text vyhledávání. Ovšem tato podpora je nabízena pouze v komerčním vyhledávacím nástroji dtSearch.
Další možnosti: je zajištěna podpora transakcí, které je možno definovat ručně a nebo nechat automaticky potvrdit každou aktualizaci. V rámci jednoho sezení je možné v danou chvíli otevřít pouze jednu transakci. Pokud je požadováno v rámci jednoho sezení otevřít sezení více, je nutné každou transakci vždy regulérně ukončit. Pouze transakce čtení není nutno potvrzovat. Tato databáze nabízí propracovaný systém tvorby záloh možností zálohování databáze za běhu nebo inkrementální zálohování (přírůstkové). Dotazování pomocí XQuery jazyka je rozšířeno o podporu SQL databází. V praxi to vypadá tak, že přístup k relačním datům je řešen pomocí tabulkového mapování do XML. Toto spojení využívá rozhraní ODBC. Možnosti bezpečnosti uživatelského přístupu a přístupových práv na úrovni databáze je řešena pro každou databázi zvlášť a implicitně je tato možnost vypnuta. Přístupová práva je možno přiřadit jak celé kolekci tak samotnému dokumentu. Jako aplikační rozhraní lze použít rozhraní pro jazyky C, Java a Scheme. Pro jazyk Java je dostupné rozhraní XML:DB a XQJ.
Nasazení: celá databáze je z pohledu provozu rozdělena na několik spolupracujících částí. Proces governor je nadřazený všem ostatním a řídí jejich koordinaci. Pokud je zaslán řídícímu procesu požadavek na spojení s databází, proces governor vytvoří proces pro sezení. Tento proces zajišťuje komunikaci s klientem, udržuje informace o spojení a řídí transakce. Každá aktivní databáze má spuštěn svůj storage manager, který drží správu nad datovými soubory.

3.4. Databáze MonetDB

Tato databáze byla poprvé vytvořena Peterem A. Bonczem a Martinem L. Kerstenem jako součást projektu MAGNUM research project na Univerzity of Amsterdam. První verze s Open-Source licencí (jako Mozilla Public Licence) byla publikována 30. září roku 2004.
V současnosti se jejím vývojem a podporou zabývá Národní ústav pro matematiku a informatiku (CWI). Návrh systému klade vysoký důraz na výkon v oblasti složitých dotazů nad rozsáhlými databázemi. Oproti tomu není kladen důraz na souběh při dotazování ani na bezpečnost.
Uložení dat: databáze používá kolekce pro rozdělení XML dokumentů. Kolekce nemají možnost jakéhokoliv dalšího zanoření. Databáze nepodporuje validaci dokumentů oproti XML schématům a ukládání dokumentů nebere ohled na schéma. Ukládání dat využívá možnosti mapování do relační databáze MonetDB. Kolekce dat je uložena jako jeden velký XML dokument, jehož kořenový element obsahuje jednotlivé XML dokumenty uložené v kolekci. Veškeré nastavení (indexy apod.) jsou nastaveny pro každou kolekci dat zvlášť. Obsah kolekce může být velmi vysoký (řádově miliony XML dokumentů). Implicitně má databáze nastaveno, aby při importu data ukládala pouze pro čtení (kvůli vybudování rychlého indexu). Index databáze vytváří znovu při každé změně v kolekci (vložení dalšího XML dokumentu apod.) Databáze nabízí možnost použití podpory aktualizací, vytváří tak jednodušší strukturu, která počítá s častějšími změnami. Logicky se výkon databáze v tomto režimu snižuje. Kolekce určené pro aktualizace automaticky zabírají v databázi více místa než kolekce určené pouze pro čtení. Databáze si musí ponechat prostor pro vyřízení pozdějších změn apod. Datový model kolekcí, které jsou aktualizovány nezachovává pořadí jednotlivých elementů.
Práce s daty: pro dotazování je standardně podporován dotazovací jazyk XQuery a jeho nadstavba XQuery Update pro aktualizace. Dotazovat lze jak samotný dokument, tak celou kolekci. S kolekcemi se může nakládat jako s množinou uzlů jednotlivých dokumentů pomocí `fn:collection` funkce nebo jako s jedním dokumentem pomocí funkce `pf:collection`. Zpracování dotazu pomocí XQuery Update Facility probíhá vždy ve vlastní transakci. Nejvyššího výkonu databáze lze dosáhnout u malých dokumentů. S rostoucí velikostí dokumentů výkon klesá. Dotazování do databáze jde řešit pomocí vlastního jazyka MonetDB (MAL, MIL, MEL).

Indexování: k indexování v této databázi dochází automaticky a není možné indexování konfigurovat. Ve všech dotazech tudíž dochází k automatickému použití dostupných indexů. V rámci použití projektu PF:Tijah je možné použít i full-text indexy .
Další možnosti: Co se transakčního zpracování týče, každý příkaz je zpracován ve vlastní transakci. Je podporován pouze tzv. auto commit mód. Z důvodu izolace všech transakcí, je možné v daném okamžiku na určitých datech provádět pouze jednu operaci z jedné transakce. Proto je doporučování ukládání více dokumentů provádět v rámci jednoho dotazu.
Nasazení: celá databáze je vystavěna na relační databázi MonetDB používající vlastní řešení v oblasti ukládání a dotazování dat. Databáze tak poskytuje rozhraní pro použití XQuery i SQL. Díky systému MMDB (main memory database), který uchovává většinu dat přímo v paměti stroje, je možné získávat výsledky i složitých dotazů velmi rychle. Paměť stroje by tedy měla být alespoň tak velká, jakou velikost mají zpracovávané dokumenty. Přidáním paměti lze částečně dotazy urychlit, neboť při zpracovávání dotazu se do hlavní paměti ukládají i mezivýsledky při zpracování dotazu. Databáze nabízí použití pouze jako samostatný server. Tento server s klientskými aplikacemi komunikuje pomocí vlastního MAPI rozhraní a také pomocí protokolu XRPC. Pro rozhraní MAPI jsou nabízeny knihovny pro jazyky jako jsou C, C++, Perl, Python, Java (pomocí JDBC ovladače). Rozhraní XRPC[21] je poskytováno zabudovaným HTTP serverem.

21 XRPC – definuje způsob zapouzdření XQuery do protokolu SOAP

4. Implementace

V rámci implementace došlo k vytvoření jednoduché aplikace, která je schopna pokládat databázi dotazy jak aktualizačního tak výběrového charakteru. Důraz byl kladen na operace vkládání, mazání, změny, výběru dat z vybraných databází. Vzhledem k masivní podpoře programovacího jazyka Java ze strany vývojářských týmů a diskuzních fór na internetu byl vybrán tento jazyk pro implementaci. Jako vývojové prostředí bylo vybráno volně dostupné prostředí NetBeans.

4.1. Popis aplikace

Vzhledem k tomu, že bylo požadováno i více vláknové testování, pro každou operaci obsahuje každá operace svojí vlastní Java třídu (XInsertThread, XUpdateThread, XDeleteThread, XQueryThread). Tato třída je v hlavní třídě pouštěné aplikace volána a je vytvořena jako nový objekt.

Příklad volání třídy

```
new Thread( new XUpdateThread(1).start() );
```

Tato třída(v tomto případě XUpdateThread) je pro každou databázi unikátní a to z důvodu častého přepisování tzv. „`connect string`“. Jedná se o řetězec, který popisuje jazyku Java, kde se daná databáze nachází. Standardně se používá jako označení stroje `localhost`(případně `127.0.0.1`). Databáze ovšem může běžet na jiném serveru než aplikace, která ji využívá.

Příklad „connect string“ databáze MonetDB

```
"Jdbc=jdbc:monetdb://localhost/db;" + "JdbcUser=monetdb;"
+ "JdbcPassword=monetdb;"
```

V aplikace je také důležitá inicializace správného ovladače (archiv tříd – zpravidla soubor typu `jar`) pro přístup do databáze. Téměř ve všech případech implementace testů docházelo k nutnosti použití jiného archivu rozhraní API pro přístup do databáze (nutnost nahrazení souboru `xmldb.jar`). Tudíž při každém testování dané databáze bylo nutné použít správného souboru tříd pro danou databázi, který je dostupný na webových stránkách společnosti, která databázi nabízí. Zdrojový kód aplikace je obsahem přílohy této práce.

4.2. Popis provozu databází

Všechny vybrané databáze byli provozovány na lokálním stroji, tudíž na stejném stroji (konfigurace: CPU Intel Pentium Dual Core – E5800 3,2 GHz, 2 GB RAM, HDD ST3250S ATA 220 GB, operační systém Windows 7 – 32 bitová verze) na kterém byla spouštěna testovací aplikace. Tímto způsobem použití odpadá nutnost zohledňovat zpoždění, které může být způsobeno použitím počítačové sítě. Všechny testované databáze byly provozovány v módu server. Kromě databází eXist a BaseX bylo použito přístupu a ovládání pomocí příkazové řádky. Databáze eXist a BaseX nabízejí možnost spravovat databáze pomocí vlastního GUI (Graphical User Interface), které je vhodné použít. Použití příkazové řádky pro 4 druhy databází je časově náročnější (nutnost častého zadávání příkazů apod.)

Příklad práce s databází Sedna

```
se_gov              – start databázového serveru
se_cdb db           – vytvoření databáze s názvem 'db'
se_sm db            – otevření databáze s názvem db 'db'
se_term – file load_data.xquery db     – import pomocí
XQuery do databáze s názvem 'db'
se_stop             – zastavení databázového serveru
```

Pro každou databázi platí jiná nastavení co se týče řízení uživatelského přístupu, řízení transakcí apod. Databáze MonetDB implicitně například ukládá data pouze pro čtení, je tedy nutné dokumenty v kolekcích ručně nastavit pro možnost změny.

Pro tyto informace je nutné procházet dostupnou dokumentaci na webových stránkách poskytovatelů databází.

5. Přehled testů

Testování jednotlivých databází bylo prováděno pomocí vlastní aplikace naprogramované v jazyku Java. Testovací aplikace je určena pro 4 základní druhy operací – operace vkládání obsahu (aktualizace), operace mazání obsahu (aktualizace), operace změny obsahu (aktualizace), operace selekce obsahu z databáze (výběr). Na testování tudíž nebyl použit žádný dostupný nástroj (XCheck, XCheck-Java, XMach-1, TPoX apod.) . Testování bylo provedeno 1, 10, 20-ti vláknově.

Testované XML dokumenty – byly použity 4 testovací XML dokumenty o různých velikostech (300MB, 600MB, 1GB, 2GB). Pro dotazování nad celou kolekcí byly použity 4 verze těchto dokumentů, přičemž každý dokument obsahoval jinou strukturu dat. Dokumenty byly vygenerovány v komerčním XML Editor programu (s darovanou studentskou licencí), který umožňuje generovat soubory s náhodným obsahem v závislosti na přiloženém XML schématu a na přiloženém XML souboru, který obsahuje slovníková data, která mohou být použita pro generování, ať již v elementech či atributech generovaného XML dokumentu. Šablonové XML schéma se stalo součástí přílohy této bakalářské práce. Všechny XML dokumenty měly strukturu viz. níže. Reálná kolekce swiss.xml obsahuje reálnou databázi proteinů obsahující proteiny nezanesené v UniProtKB/Swiss-Prot. Databáze obsahuje i duplicity.

Informace o testovaných souborech

jméno dokumentu	velikost (MB megabyte)	velikost (počet elementů)
instance300m.xml	300	200 tis.
instance600m.xml	600	400 tis.
instance1g.xml	1000	680 tis.
instance2g.xml	2000	1360 tis.
swiss.xml (reálná kolekce)	716	26,5 tis.

Příklad struktury testovaného dokumentu

```
<zamestnanci>
	<zamestnanec id="1809235589">
		<jmeno>Eva</jmeno>
		<prijmeni>Termoska</prijmeni>
		<adresa>adresa0</adresa>
		<email>card@erste.com</email>
		<web>www.java.cz</web>
		<plat>38459043</plat>
		<narozen>2006-05-04</narozen>
	</zamestnanec>
</zamestnanci>
```

5.1. Operace aktualizace databáze

V této části dojde k popisu jednotlivých operací, které databáze provádí při spuštění aktualizačního dotazu. Primárně se zaměříme na operace vložení obsahu do již uloženého dokumentu v kolekci databáze. Dále si popíšeme možnosti výmazu obsahu již uloženého dokumentu. V třetí části dojde k popisu operace, která provádí změnu v datech v dokumentu. V rámci testování bylo vypuštěno měření veličin při ukládání a mazání celých dokumentů z databáze. Tyto operace byly časově velmi náročné a toto měření není tématem této práce.

5.1.1. Operace vkládání dat

eXist, BaseX, Sedna – XUpdate: pro vkládání do těchto tří databází bylo použito rozhraní jazyka XUpdate pro jazyk Java, s kterým tyto databáze umí pracovat.

Příklad dotazu INSERT v jazyce XUpdate

```
<xupdate:append select="/zamestnanci">
	<xupdate:element name="zamestnanec">
	<xupdate:attribute name="id"> number
</xupdate:attribute>
	<jmeno> name X</jmeno>
	<prijmeni> surnameX </prijmeni>
	<email> Xemail@email.com </email>
	</xupdate:element>
</xupdate:append>
```

MonetDB – XQuery Update: vkládání elementů v případě této databáze probíhalo pomocí rozhraní jazyka XQuery Update.

Příklad dotazu INSERT v jazyce XQuery Update

```
do insert
      <jmeno> nameX </jmeno>
      <prijmeni> surnameX </prijmeni>
      <email> Xemail@email.com </email>
after fn:doc ('persons')/zamestnanci/zamestnanec[1]
```

5.1.2. Operace mazání dat

eXist, BaseX, Sedna - XUpdate: výmaz obsahu z těchto databází bylo realizováno aplikačním rozhraní, které používá dotazovací jazyk XUpdate.

Příklad dotazu DELETE v jazyku XUpdate

```
<xupdate:remove
select="/zamestnanci/zamestnanec[@id='numberX']">
</xupdate:remove>
```

MonetDB – XQuery Update: k provedení operace mazání obsahu dokumentu byl použit jazyk XQuery Update, jehož aplikační rozhraní bylo použito ve vlastní Java aplikaci

Příklad dotazu DELETE v jazyku XQuery Update

```
do delete fn:doc
('instance300m.xml')/zamestnanci/zamestnanec['numberX']
```

5.1.3. Operace změny dat

eXist, BaseX, Sedna - XUpdate: pro změnu obsahu těchto databází byl zvolen jednoduchý XUpdate dotaz pro změnu tolika elementů, jak velký byl cyklus v aplikaci (v našem případě 1000 průchodů).

Příklad použití dotazu UPDATE v jazyku XUpdate

```
<xupdate:update
select=//zamestnanci/zamestnanec[@id=numberX]/jmeno>JmenoX
</xupdate:update>
```

MonetDB - XQuery Update: zvolen jednoduchý dotaz XQuery Update, který je ekvivalentní s přechozími použitími.

Příklad použití dotazu UPDATE v jazyku XQuery Update

```
replace value of node
//zamestnanci/zamestnanec[numberX]/jmeno with "JmenoX"
```

5.2. Operace pro výběr dat z databáze

5.2.1. Dotazy na shodu

Dotazem na rovnost je myšlena přesná shoda hledaného slova s hodnotou daného elementu (v relačních databázích použité jako WHERE = ‘výraz’).

Příklad použití dotazu SELECT v jazyku XQuery

```
//zamestnanci/zamestnanec/jmeno='Jiri'
```

5.2.2. Dotazy na počátek

Tímto dotazem se budeme dotazovat na data, která se podobají hledanému výrazu (v relačních databázích známé pod klauzulí WHERE sloupec LIKE = ‘%výraz%’).

Příklad použití dotazu SELECT v jazyku XQuery

```
for $i in //zamestnanci/zamestnanec/jmeno
where starts-with($i, 'Jir')
return $i
```

5.2.3. Full textové vyhledávání

Příklad použití dotazu full textového vyhledávání

```
//refinfo [title ftcontains "peptides"]
```

6. Výsledky testů

Testy byly zaměřeny zejména na čas provedení jednotlivých dotazů. Výsledný čas provedení je ve všech případech uváděn v milisekundách. Během testů bylo přihlíženo na další charakteristiky přístupu k datům zejména na IO přístupy na pevný disk ve formě čtení/zápisu na disk. Výsledky tohoto měření jsou uváděny v MB/s (megabyte za sekundu). Statistiky přístupu na disk (charakteristika, která je většinou měřena u relačních databází) provedené během jedné operace nebyl měřen vzhledem k použitému hardwaru pro testování. Zdrojem měřených hodnot se stal především nástroj dostupný na operačních systémech založených na platformě Windows – program Perfmon (Performance monitor – spuštění pomocí příkazu '`perfmon.exe`').

6.1. Testování aktualizace databáze

Ve výsledných grafech jsou zohledněny výsledky, které byly dosaženy při běhu aplikace jedno, deseti a dvaceti vláknově. Aplikace tak je schopna simulovat provoz běžné aplikace, kterou používá současně několik uživatelů

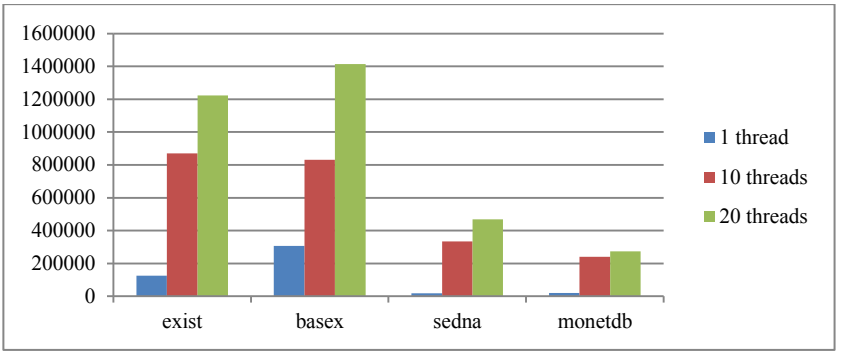
6.1.1. Operace vkládání obsahu do XML dokumentu

Insert - Tato část programu vkládá do XML dokumentu vloženého do kolekce pomocí 1, 10, 20 vláken. Je tak vykonáváno tisíc a 10 tisíc operací.

Tabulka hodnot

db	1 vlákno	10 vláken	20 vláken
eXist	125702	870720	1223013
BaseX	307062	830784	1414540
Sedna	17820	334500	469202
MonetDB	20144	240229	272762

Výsledky – graf



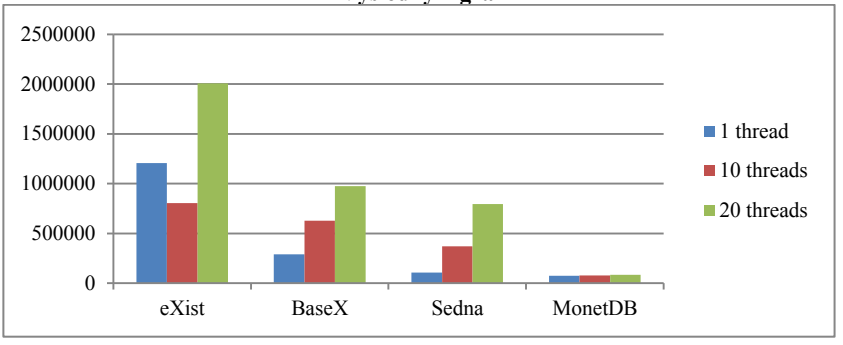
6.1.2. Operace mazání obsahu z XML dokumentu

Delete – v tomto testu probíhalo mazání 1 tisíce vložených a změněných elementů v kapitolách 5.1.1. a 5.1.3.

Tabulka hodnot

db	1 vlákno	10 vláken	20 vláken
eXist	1206942	804230	2008332
BaseX	289921	627104	973263
Sedna	105100	370850	794020
MonetDB	73210	78229	83886

Výsledky – graf



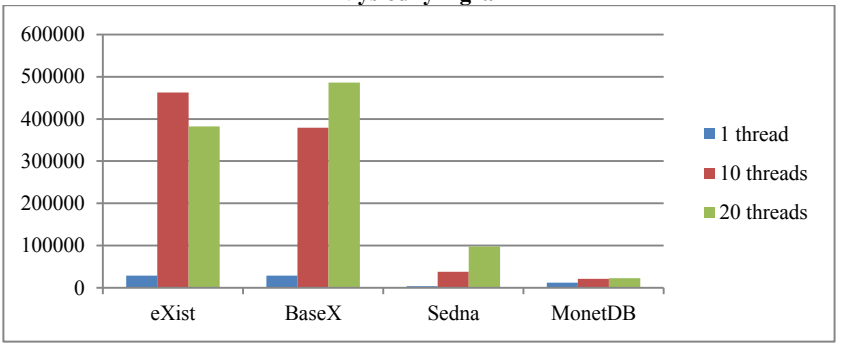
6.1.3. Operace změny obsahu XML dokumentu

Update – tento test provádí aktualizaci (update) 1 tisíce elementů vložených v testu, který je veden pod kapitolou 5.1.1.

Tabulka hodnot

db	1 vlákno	10 vláken	20 vláken
eXist	28934	462331	382004
BaseX	28933	379100	486150
Sedna	3804	38051	98011
MonetDB	12205	21227	23002

Výsledky – graf



6.2. Testování výběru dat z databáze

Testování v tomto případě proběhlo jak nad jednotlivými dokumenty, tak nad celou kolekcí XML dokumentů. Mezi těmito případy už je vidět poměrně markantní rozdíl ve výsledném čase, který byl třeba pro dotazování vyhradit.

6.2.1. Dotaz na výběr dat z XML dokumentu v kolekci

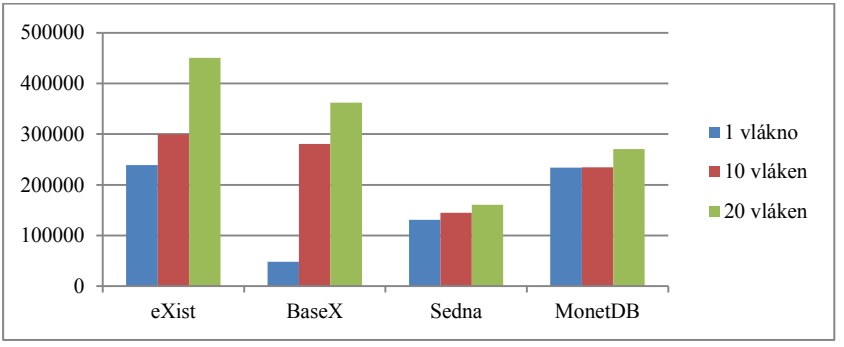
6.2.1.1. Dotaz na ekvivalenci

Tabulka hodnot

db	dotaz na ekvivalenci		
	1 vlákno	10 vláken	20 vláken

eXist	239220	299600	450788
BaseX	47850	280603	362007
Sedna	130862	145063	160741
MonetDB	233721	234440	270770

Výsledky – graf

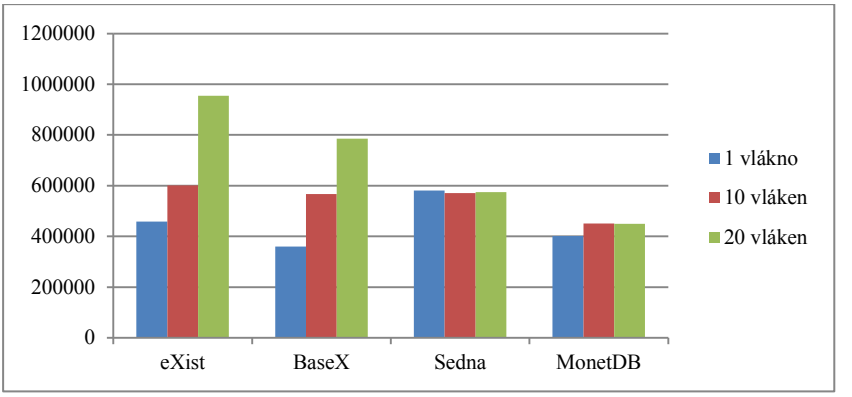


6.2.1.2. Dotaz na podobnost

Tabulka hodnot

db	dotaz na podobnost		
	1 vlákno	10 vláken	20 vláken
eXist	458900	600785	954006
BaseX	360100	566880	785002
Sedna	580922	570850	574447
MonetDB	401760	450778	449012

Výsledky graf



6.2.2. Dotazy na výběr dat z reálné kolekce

6.2.2.1. Dotaz na ekvivalenci

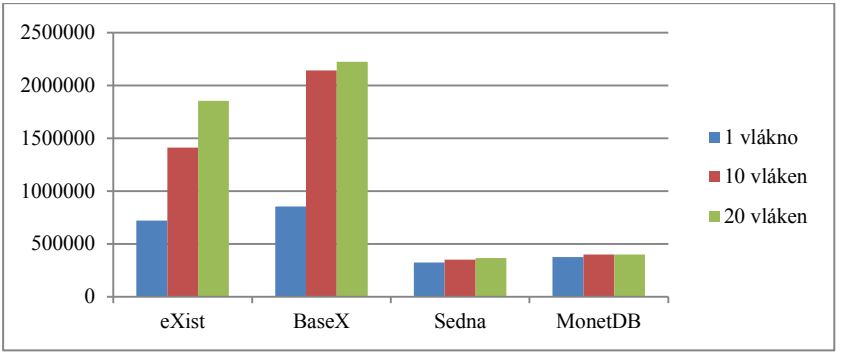
Příklad použití dotazu SELECT v jazyku XQuery

```
//refinfo/authors/author='Smith, E.L.'
```

Tabulka hodnot

db	dotaz na ekvivalenci		
	1 vlákno	10 vláken	20 vláken
eXist	721450	1411007	1854010
BaseX	855200	2140741	2224560
Sedna	322533	352400	365410
MonetDB	375500	400022	399410

Výsledky – graf



6.2.2.2. Dotaz na podobnost

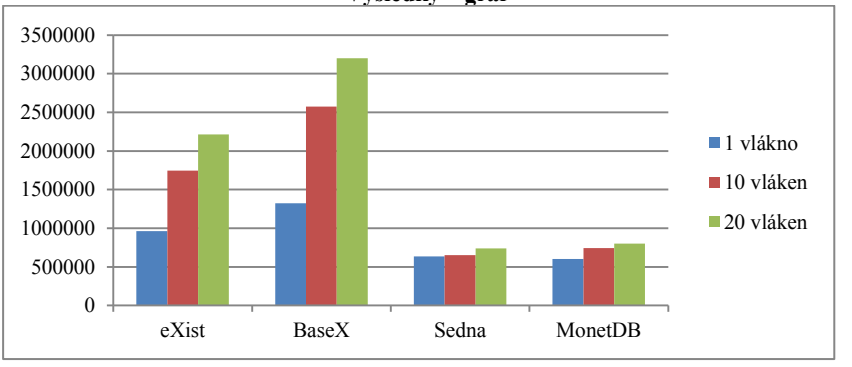
Příklad použití dotazu SELECT v jazyku XQuery

```
for $i in //refinfo/authors/author
where starts-with($i, 'S')
return $i
```

Tabulka hodnot

db	dotaz na podobnost		
	1 vlákno	10 vláken	20 vláken
eXist	963253	1744149	2214600
BaseX	1321004	2574852	3200920
Sedna	633262	652100	740156
MonetDB	601421	741008	800748

Výsledky – graf



6.2.2.3. Full textové vyhledávání

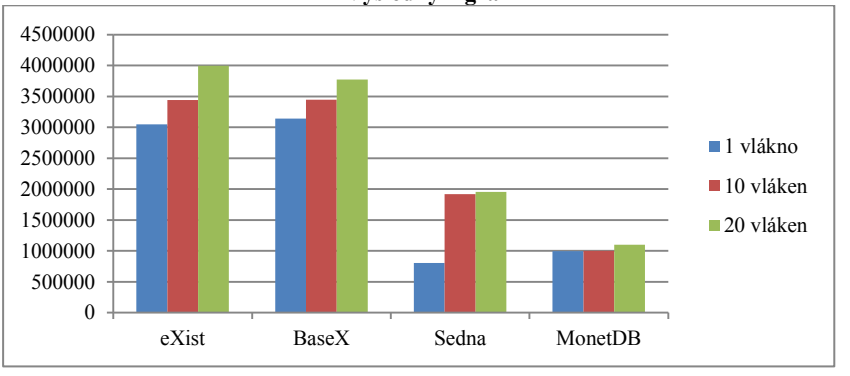
Příklad použití dotazu SELECT v jazyku XQuery

```
//refinfo [title ftcontains "peptides"]
```

Tabulka hodnot

db	full-text dotaz		
	1 vlákno	10 vláken	20 vláken
eXist	3045200	3441002	3996200
BaseX	3141260	3447852	3774170
Sedna	804706	1920669	1955000
MonetDB	995262	1002150	1102360

Výsledky – graf



7. Závěr

Ze získaných výsledků je zřejmé, že pro velké objemy dat (stovky megabajtů až jednotky gigabajtů) se stále v horní části žebříčku Open-Source řešení drží databáze MonetDB. Jako velmi výkonná se dle výsledků ukázala i databáze Sedna, která ovšem na poli full textového vyhledávání vykazuje delší odezvy, než s kterými by bylo možno počítat při návrhu aplikace, která s full textovým prohledáváním počítá. V této databázi je ovšem možno použít komerční (placený) modul dtSearch pro zlepšení výkonu této funkcionality. Podobnou vlastnost nabízí i databáze MonetDB ve formě modulu PF:Tijah, jejíž otestování bohužel z časových důvodů nebylo provedeno.
Zbývající dvě databáze (eXist, BaseX) nijak zvlášť nevynikaly svojí výkonností. Pozitivní na těchto databázích je ovšem jejich uživatelsky přívětivé ovládání a především sofistikovaně řešené GUI, které nabízí komfortní ovládání celé databáze.
Co se Nativních XML databází týče, je nutno podotknout, že jsou na trhu IT řešení považovány za poměrně mladou technologií a jsou stále ve vývoji. Oproti relačním databázím jsou stále ještě řešeny aspekty jako je indexování, jednotná podpora dotazovacích jazyků apod. Nativní XML databáze se jeví jako vhodné řešení pro aplikace, jejichž výsledky dotazů jsou zobrazovány popřípadě zpracovávány v XML formátu (XHTML apod.).

8. Použitá literatura

1. rpbourret.com - XML and Databases. [Online] 2010. [Datum 25. července 2012] http://www.rpbourret.com/xml/XMLAndDatabases.htm

2. rpbourret.com - XML Database Products. [Online] 20. června 2010. [Datum 25. července 2010] http://www.rpbourret.com/xml/XMLDatabaseProds.htm#native

3. datakon.cz (Jaroslav Pokorný, MFF UK Praha) - XML databáze: současný stav a perspektivy. [Online] 2004. [Datum 26. července 2012] http://www.datakon.cz/datakon08/d04_it_pokorny.pdf

4. jboss.org - JBoss Documentation. [Online] 2012. [Datum 30. července 2012] https://access.redhat.com/knowledge/docs/en-US/JBoss_Enterprise_Application_Platform/6/html/Beta_Documentation/index.html

5. xmldb-org.sourceforge.net - XML:DB Initiative for XML Database. [Online] 2000-2003. [Datum 2. srpna 2012] http://xmldb-org.sourceforge.net/index.html

6. rpbourret.com - XML-Enabled Databases. [Online] 2010. [Datum 2. srpna 2012] http://www.rpbourret.com/xml/ProdsXMLEnabled.htm

7. w3.org - XML Path Language (XPath), XSL Transformations (XSLT). [Online] 16. listopadu 1999. [Datum 2. srpna 2012] http://www.w3.org/TR/

8. sourceforge.net - Ozone - Java OODBMS. [Online] 28. října 2010. [Datum 2. srpna 2012] http://sourceforge.net/projects/ozone/

9. w3.org - W3C XML Query (XQuery). [Online] 7. května 2012. [Datum 2. srpna 2012] http://www.w3.org/XML/Query/

10. xmldb-org.sourceforge.net - XUpdate. [Online] 14. září 2000. [Datum 2. srpna 2012] http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html

11. w3.org - XQuery Update Facility 1.0. [Online] 17. března 2011. [Datum 2. srpna 2012] http://www.w3.org/TR/xquery-update-10/

12. basex.org - BaseX. The XML Database. [Online] 2012. [Datum 2. srpna 2012] http://basex.org

13. oracle.com - Oracle Berkeley DB XML. [Online] 2012. [Datum 2. srpna 2012]

http://www.oracle.com/technetwork/products/berkeleydb/index-083851.html

14. exist-db.org - eXist-db Open Source Native XML Database. [Online] 3. července 2012. [Datum 2. srpna 2012]
http://exist-db.org/exist/index.xml

15. monetdb.org - MonetDB. [Online] 2012. [Datum 2. srpna 2012]
http://www.monetdb.org/Documentation

16. xml.apache.org - Apache Xindice. [Online] 2012. [Datum 2. srpna 2012]
http://xml.apache.org/xindice/

17. sedna.org - Sedna Native XML Database System. [Online] 28. listopadu 2011. [Datum 2. srpna 2012]
http://www.sedna.org/documentation.html

18. sourceforge.net - dbXML. [Online] 17. září 2009. [Datum 2. srpna 2012]
http://sourceforge.net/projects/dbxml-core/

19. otago.ourarchive.ac.nz - Performance of Relational Databases versus Native Databases, Anne Williams. [Online] 4. října 2005. [Datum 2. srpna 2012]
http://otago.ourarchive.ac.nz/bitstream/handle/10523/1200/AnneWilliamsOCR.pdf

20. nativexmldatabase.com - XML versus Relational Database Performance. [Online] 22. srpna 2010. [Datum 2. srpna 2012]
http://nativexmldatabase.com/2010/08/22/xml-versus-relational-database-performance/

9. Přílohy

číslo	jméno souboru	obsah souboru	úložiště
I	final_results.xlsx	Výsledky testování	Edison, DVD
II	used_queries.xqeury	Použité dotazy v testování	Edison, DVD
III	java_project.zip	Zdrojové soubory testovací aplikace	Edison, DVD
IV	xml_doc_structure.xml	Struktura generovaných XML souborů.	Edison, DVD
V	xml_files.zip	Vygenerované soubory	DVD
VI	xml_real_coll.zip	Reálná kolekce	DVD